

PATENT ABSTRACTS OF JAPAN

(11) Publication number : 08-115246
 (43) Date of publication of application : 07. 05. 1996

(51) Int. Cl. G06F 12/00
 G06F 13/00
 G06F 17/30

(21) Application number : 07-244973 (71) Applicant : XEROX CORP
 (22) Date of filing : 22. 09. 1995 (72) Inventor : TERRY DOUGLAS B
 DEMERS ALAN J
 PETERSEN KARIN
 SPREITZER MICHAEL J
 THEIMER MARVIN M
 WELCH BRENT B

(30) Priority

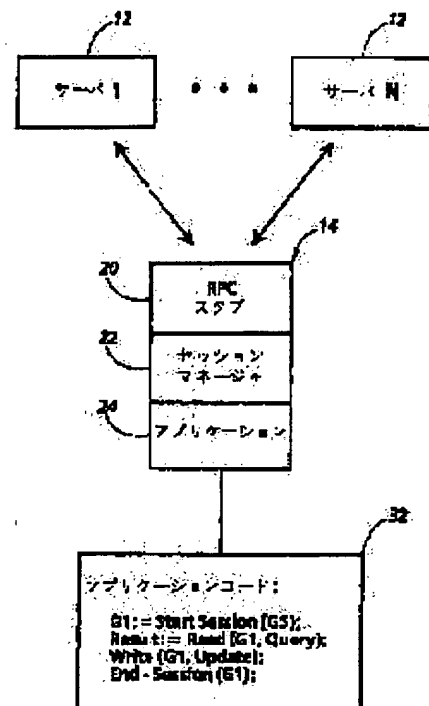
Priority number : 94 314971 Priority date : 28. 09. 1994 Priority country : US

(54) METHOD FOR GUARANTEEING CONSISTENCY OF SESSION

(57) Abstract:

PROBLEM TO BE SOLVED: To guarantee the consistency selected by the client of a duplicate data base having weak consistency to the client at every 'session'.

SOLUTION: A plurality of sessions which are respectively constituted of data base access and updating requests from at least one client are defined and, when the propriety of arbitrary guarantee is decided by at least one client, a set of the related parts of the related consistency guarantee of each session is selected from a prescribed set of guarantee. Even when at least one client performs access to different servers during a session, the selective guarantee of the client is executed during the session.



LEGAL STATUS

[Date of request for examination] 24. 09. 2002
 [Date of sending the examiner's decision]

of rejection]

[Kind of final disposal of application
other than the examiner's decision of
rejection or application converted
registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's
decision of rejection]

[Date of requesting appeal against
examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998, 2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-115246

(43) 公開日 平成8年(1996)5月7日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 12/00	5 3 3 F	7623-5B		
13/00	3 5 7 Z	7368-5E		
17/30				
		9194-5L	G 0 6 F 15/ 40	3 1 0 C

審査請求 未請求 請求項の数1 O L (全 17 頁)

(21) 出願番号 特願平7-244973

(22) 出願日 平成7年(1995)9月22日

(31) 優先権主張番号 3 1 4 9 7 1

(32) 優先日 1994年9月28日

(33) 優先権主張国 米国 (U S)

(71) 出願人 590000798

ゼロックス コーポレーション
XEROX CORPORATION
アメリカ合衆国 ニューヨーク州 14644
ロチェスター ゼロックス スクエア
(番地なし)

(72) 発明者 ダグラス・ビー・テリー

アメリカ合衆国 カリフォルニア州
94070 サンカルロス ビバリードライブ
240

(74) 代理人 弁理士 小堀 益 (外1名)

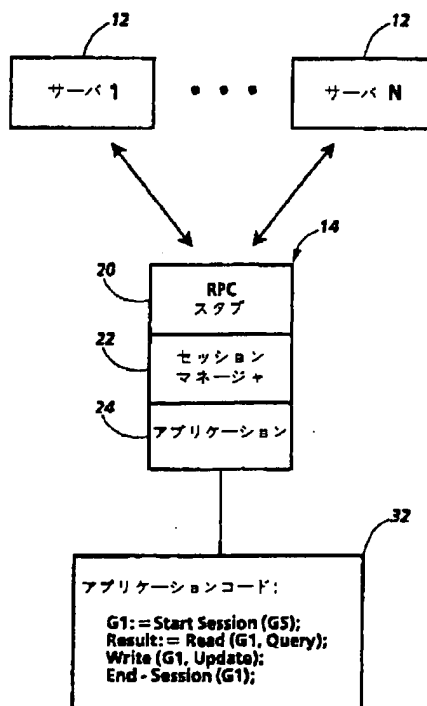
最終頁に続く

(54) 【発明の名称】 セッションの一貫性を保証する方法

(57) 【要約】

【課題】 弱一貫性の複製データベースのクライアントに対して、「セッション」ごとに、クライアントの選択した一貫性保証を提供する。

【解決手段】 少なくとも一台のクライアントのデータベースアクセスおよび更新要求から各々が構成されているような複数のセッションを定義し、前記少なくとも一台のクライアントによって、任意の保証の適切さが決定された場合に、所定の保証の集合から、各セッションの関連一貫性保証の関連部分集合を選択し、少なくとも一台のクライアントがセッション中に異なるサーバにアクセスするときにおいても、前記セッションの最中に前記少なくとも一台のクライアントの選択保証を実施する。



【特許請求の範囲】

【請求項1】 複数のサーバと複数のクライアントを有し、前記クライアントのうちの少なくともいくつかは、前記サーバが不一致なデータ値を包含する時間を含む種々の時間に種々のサーバにアクセスすることがあり、前記サーバは弱一貫性の複製データを保存し、前記データは前記クライアントによりアクセス可能で且つ更新可能であり、前記データは、更新されると、前記更新データを有するサーバによって前記更新データを有しないサーバに伝播されるデータベースシステムで、各々のセッションに属するアクセスまたは更新要求に対して、毎セッションベースで、いずれかセッションに参加しているあらゆるクライアントに対して、前記データの関連一貫性保証を提供するための方法であって、

A) 少なくとも一台のクライアントのデータベースアクセスおよび更新要求から各々が構成されているような複数のセッションを定義するステップと、

B) 前記少なくとも一台のクライアントによって、任意の保証の適切さが決定された場合に、所定の保証の集合から、各セッションの関連一貫性保証の関連部分集合を選択するステップと、

C) 少なくとも一台のクライアントがセッション中に異なるサーバにアクセスするときにおいても、前記セッションの最中に前記少なくとも一台のクライアントの選択保証を実施するステップ、とから成る、前記方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、分散形データベースのクライアントにデータの一貫性(consistency)を保証するための方法に関するものであり、更に詳しくいうと、セッションの間中、クライアントに一貫性を保証するための方法に関するものである。

【0002】

【従来の技術】 多数のユーザの間のデータへの共有アクセスを提供するシステムにおいて、一般的に何らかの形でデータの「複製」が行われる。例えば、システムによっては、各々がシステムデータベースの完全な複製を有する多数のデータ「サーバ」に分割されるものもある。これらサーバは、多数の「クライアント」からの要求を処理する。一般に、かかる要求は、システムのベースの「読出し」と「書込み」という形で入ってくる。

【0003】 データの複製は、移動性のあるクライアントに対して柔軟性と可用性を与えるものである。移動性のあるクライアントは、さき接続していたものと違うサーバに後で接続するためだけに、断続的にシステムを切断する。全部のサーバでデータが完全に(そして完璧に)複製されていれば、移動性のあるクライアントは、少なくとも一つのサーバに接続できさえすれば、常にそこで全部のシステムデータにアクセスするであろう。

【0004】

【発明が解決しようとする課題】 しかしながら、特に多数のクライアントとサーバを有するシステムでは、「完璧な」複製は一般に非実際的なすなわち極端に費用を要する。クライアントによってもたらされる(書込みによる)変更は、完璧な複製を保証するために、必ず全部のシステムサーバに速やかに伝播される必要がある。そうでなければ、クライアントの中に、まだ更新されていない、すなわち複製の完璧性に反するサーバのデータ項目にアクセスするものが出る可能性がある。従って、この条件を緩和するための別の計画が現れた。

【0005】 「強一貫性」の分散形データベースは、変更事項がシステム中くまなく更新されること、または、不一致が生じた場合にある種のサーバ起動コールバックを行うこと、が要求される。強一貫性のシステムは高度なデータ一貫性を提供するが、しばしば代償となる犠牲は可用性である。例えば、このようなデータベースのクライアントは、更新されたばかりで全部のサーバ中にデータの更新が行われるまでブロックされているデータにアクセスする場合があるかも知れない。

【0006】 一方、「弱」一貫性のデータベースでは、サーバは、全部のシステム更新内容を含んでいないかも知れないデータにアクセスを許される。かかるシステムは、一般に、サーバ間の更新内容の「遅延」伝播(すなわち、更新内容が時間をかけてサーバ間に伝播する)を特徴とする。従ってクライアントは、別の複製のデータを読んだときに不一致の価値を見出す。それにも関わらず、弱一貫性のシステムは、その高い可用性、優れたスケーラビリティ、およびデザインの単純さにより、人気がある。これらの利益は、複製間の同期化がほとんど或は全くない状態で、読出しと書込みが実施されるようにする能力により生じたものである。

【0007】 残念なことに、弱一貫性のシステムの読み書き動作の配列に関する保証がないため、ユーザならびにアプリケーション業務を混乱させる可能性がある。ユーザは、あるデータ項目のいずれかの内容を読み、後から、古いほうの内容を読むことがあるかも知れない。同様に、あるユーザが、いずれか別のデータの読出しに基づいてデータ項目を更新しても、別の人は、その基礎となったデータを見ずに、更新された項目だけを読む場合がある。弱一貫性のシステムの重大問題は、たった一人のユーザまたはアプリケーション業務がデータ変更を行っているときにおいても、不一致の生じる可能性があることである。例えば、分散形データベースシステム中の移動性のあるクライアントは、あるサーバで書込みを行ない、後から別のサーバで読出しを行うことがある。二作業間のあいだに二つのサーバが互いに同期化されなければ、クライアントは不一致な結果を見ることになる。

【0008】 データの利用可能性ならびにデータの一貫性は、相互排他的な目標であることがあるので、システムデザイナー達は、歩み寄りの道をとろうと努力してき

た。しかしながら、システムは一般に、各自が別々のセキュリティと利用可能性のニーズを有する莫大数のクライアントをサポートしているため、このトレードオフに対するシステム基模で課せられる解決は、クライアントの将来展望という点では最適状態に及ばない。クライアントの中には、利用可能性を損なっても更なるデータ一貫性を欲するものもいるが、しばしば不一致バージョンを入手することを意味するとしても、複数のデータコピーにアクセスすることを好むものもいる。

【0009】従って、複製データベースのクライアントが、適切な、個別決定された、自分自身のためのデータ一貫性とデータ利用可能性のバランスを確立できるようにする機構を提供する必要がある。

【0010】

【課題を解決するための手段】本発明は、弱一貫性の(weakly consistent)複製データベースのクライアントに対して、「セッション」ごとに、クライアントの選択した一貫性保証を、提供するものである。「セッション」とは、論理的に関連付けられる、データベースに対する読出しと書き込みの順序と広義に定義される。セッションは、同時に存在する他のセッションとは実質的に独立に開始および終了される。しかしながらクライアントは、クローンセッションおよび/またはマージセッションが許可される。

【0011】

【発明の実施の形態】図1は、クライアント/サーバ環境の図である。図2は、クライアントとサーバとセッションマネージャの構成の一例である。図3は、クライアントの申し込みによって引き起こされた単純なセッションの一例を示す。図4と5は、各々、「自分の書き込み読出し」保証が不在の場合と存在する場合のクライアントトランザクションの一例を示す。図6と7は、各々、「単調読出し」保証が課せられない場合と課せられる場合のクライアントトランザクションを示す。図8と9は、各々、「読出し後書き込み」保証が課せられない場合と課せられる場合のクライアントトランザクションを示す。図10と11は、各々、「単調書き込み」保証が課せられない場合と課せられる場合のクライアントトランザクションを示す。図12と13は、セッションマネージャによって実施される発見ならびに実施プロセスのフローチャートである。図14は、セッションマネージャのブロック図である。図15は、サーバは十分に最新情報に接しているかを判定するバージョンベクトルの使用法を示す。

【0012】ここで図1を参照すると、普通の「クライアント/サーバ」モデル10において、サーバ12は種々クライアント14によって作成された、または、他のサーバからコピーされたデータの陳列所(すなわちデータベース)である。文中、「データベース」という用語は、特定のデータモデルまたは組織を意味するものでは

ない。データベースは、単にデータ項目の集合であり、データ項目というのは、従来のファイルから関連データベース中の集合までのいずれかである。

【0013】サーバ12は、他のサーバ12および/または一人または複数のクライアント14と通信中であるように図示されている。これらの通信回線は、連続している場合もあるし、時々とぎれる場合もある。クライアントとサーバ間の最も一般的なやりとりは、種々のサーバ12に保存されているデータに対する「読出し」または「書き込み」要求の形で発生する。「読出し」または「書き込み」要求は、データベースに保存されている特定データ項目に対して指定的なものである場合もあるし、あるいは、データベース照会の形を取る場合もある。例えば、「読出し」は、「特定ファイルの中身をリターンせよ」というような単純な検索回収作業の場合もあるし、「100,000ドル以上になった全従業員の氏名をリターンせよ」というような複雑な照会の場合もある。書き込み操作はデータベースを更新する。一方、「書き込み」は、データ項目の作成、変更、削除に関わる場合もある。実際のところ、書き込みは、サーバのデータベースの複数の項目を最小限更新するトランザクションを表す場合さえある。

【0014】時間依存成分を有するので、サーバのデータベースを見ることは、しばしば役に立つ。従って、「DB(S, t)」は、時間t以前にサーバSに受信された、順序付けられた「書き込み」配列であると定義される。tが現在の時間である場合には、それは、DB(S)として書き込まれ、サーバのデータベースの現在の内容を表す。概念的にサーバSは、空のデータベースで初めて所定順序でDB(S)の各書き込みを適用することによって、自身のデータベースのコピーを作成する。実際的には、サーバは、結果として得られたデータベースのエントリの順序が変更されないでいる限り、別の順序で書き込みを処理できる。従って、例えば、DB(S)の書き込みの順序は、必ずしもサーバSが最初に書き込みを受信した順序とは一致しない。

【0015】アップデートは時間をかけて(極めて小さくでもなく、瞬時にでもなく)徐々に普及するので、モデル10のデータベースは「弱一貫性」である。弱一貫性であるため、サーバが異なるとデータベースのコピーが変化する。すなわち、二台のサーバS1とS2について、DB(S1, t)とDB(S2, t)は必ずしも等しくない。しかしながら、アップデートが無い状態ではサーバは全く同じデータベースへ収束するので「究極的一貫性」が存在する。言い換えると、各々の書き込みWについて、どのサーバSのDB(S, t)にもWが存在する時間tが存在する。偶発的一貫性は、「全伝播」(すなわち、更新された全データが最後に各サーバに受信される場合)と「一貫した順序付け」(すなわち、「順序依存性」を有する書き込みが、全部のサーバで同様に順序

付けられた場合)という二特性によって決まる。

【0016】究極的一貫性を得るために、モデル10は、種々サーバをわたるデータの複製について二つの仮定を立てる。まず、モデル10は、複製が「遅延」伝播によって行われると仮定する。遅延伝播は、各々の書込みが時間をかけて他のサーバに伝播されるものである。第二に、モデル10は、全部のサーバは全部の「非可変」書込み(すなわち、データベースへの適用の「順序」により、異なる二つのデータベースになる書込み)を同じ順序でデータベースに適用する、と仮定する。そのため、「書込み順序(W1, W2)」は、書込みW2の前に書込みW1が順序付けられているか否かを示す論理術語として定義される。このように、書込み順序(W1, W2)が真であれば、W1は、W1とW2の両方を受信したいずれかサーバのDB(S)で、W2の前に順序付けられるはずである。

【0017】究極的一貫システムでは、サーバは、書込みの順序に同意する(従って、論理書込み順序(W1, W2)が満足されているか検査もする)ための種々技術のいずれかを利用できる。かかる技術の一つでは、書込みは、提示される書込み要求の順序に基づいてサーバが与える「発信タイムスタンプ」によって順序付けられる。しかしながら、書込みが実施された実際の時間によって書込みを順序付けるという要求がないので、書込み順序を決定するためにタイムスタンプを利用することはサーバがクロックに同期していたことを意味するものではない。

【0018】結論として、弱一貫性のシステムは矛盾のある書込みを生じさせる。すなわち、二台のクライアントが同じデータ項目に対して同時的な両立し得ない更新を行うことがある。既存のシステムは、いろいろな方法で矛盾のある書込みを解決する。システムの中には、書込み順序によって、どの書込みが最初に命ぜられたかを判断するものもあるし、検出された矛盾の解決を人間に頼るものもある。

【0019】図2で、矢印の付いた線は、クライアント、サーバ、セッションマネージャとして識別される実体の、説明に役立つ通信経路を表す。これらの実体は、同じもしくは異なるホストに常駐する。その場合、最適なのは別々の通信手段にすることである。例えば、ホストの同じアドレス空間に二つの実体が常駐する場合は、単純な手続き呼び出しが適切な通信手段であるのに対し、二つの実体が異なるホストに常駐する場合は遠隔手順呼び出し(RPC)が最適である。

【0020】クライアント14は、セッション操作の一部として、セッションマネージャ22とやりとりする。同一セッションに複数のクライアントが属する場合もあるし、各々のクライアントが複数のセッションに参加する場合もある。いずれか任意のセッションは、一つのセッションマネージャによって操作されるが、一つのセ

ッションマネージャが複数のセッションを操作する場合もある。セッションマネージャ22は、クライアント14に代わってサーバとやりとりする。サーバは自分たち同士でやりとりを行って、更新を伝播する。

【0021】各セッションは、セッション識別子が付随する。セッション識別子は、セッションマネージャによって異なるセッションを区別するために使用される。同一セッション識別子を使用することにより、複数クライアントが一つのセッションに参加することができる。従って、クライアントは、共有したいセッションのセッション識別子を交換できなくてはならない。

【0022】セッションマネージャ22は、セッションの初期設定、セッション毎のデータ保証集合の選択、セッション毎のこれら保証の実施、を処理する。サーバ12からのデータベースサービスは、一般的にはクライアント14から要求される。クライアントはセッションの一部として要求を行う。セッションの一貫性保証を実施するために、クライアントは、そのセッションに対して応答可能なセッションマネージャ22にSの要求を行う。セッションマネージャは適当なサーバとやりとりして、クライアントのデータベース要求を実施する。

【0023】クライアントとセッションマネージャが、セッションならびにそれらの保証を実施するので、サーバは一般に、どのセッションが活動中であるかを知らない。サーバ12は、要求を満足するデータを、そのデータベース30から検索回収した後、それを要求元にリターンすることによって、読出し要求を処理する。書込み要求は一般に、そのデータベースの内容を変更することによって処理される。

【0024】この特定実施例を実施するには、各々のサーバは、自分の現在の状態を記憶する必要がある。大まかに言うと、サーバの「状態」は、サーバが「知っている」全書込みの集合のことである。この書込み情報の細分性は、データの大きさによって異なる。しかしながら、書込み情報の細分化が細かいほど、保証実施の諸経費が大きくなる。

【0025】この状態情報の目的は、サーバのデータベースが十分に「最新な」ものであるか否かを判断し、一貫性保証の実施を容易にすることである。到達可能なずれのサーバのデータベースも十分に「最新な」ものでなかった場合には、セッションマネージャはクライアントに通知を行う。大まかな意味で、「セッション」とは、一台または複数台のクライアントによって出された一連の読出しおよび書込み要求である。このように、クライアントは、セッションの開始とその持続時間の制御に多大な自由裁量を与えられる。更に、下記の基本命令により、更なる機能性も提供される。

【0026】

【表1】

表1-セッション基本命令

START_SESSION(Guarantee_Set)

/* クライアントのアプリケーション業務はSTART_SESSION
を呼び出し、セッションの最中に維持されるべき保証の集合
をセッションマネージャに渡す。/*

SID := Allocate_State();

/* セッションマネージャは、「セッション状態」として維持さ
れるデータ構造 Read_Set("RS")と Write_Set("WS")を作
る。/*

SID.GS := Guarantee_Set;

SID.RS := {}; /* 読出し集合をヌルに初期設定/*

SID.WS := {};

Return SID; /* 「セッションID」(SID)を呼び出しアプリケーション
業務にリターンする。/*

END_START_SESSION

END_SESSION

/* アプリケーション業務は、セッションIDで指定されるセッ
ションを終了する/*

Deallocate(SID); /* セッション状態を再請求することによって終
了/*

END_END_SESSION

【表2】

表2

```

DOWNGRADE_GUARANTEES(SID, Deleted_Guarantees);
/* セッション中の保証をクライアントが格下げ出来るようにす
   る。 */
SID.GS := SID.GS - Deleted_Guarantees;
END_DOWNGRADE_GUARANTEES

```

```

CLONE_SESSION(SID1)
/* クライアントがセッションを「コピー作成」できるようにす
   る。 */
SID2 := Allocate_State();
SID2.GS := SID1.GS
SID2.RS := SID1.RS;
SID2.WS := SID1.WS;
Return SID2;
END_CLONE_SESSION

```

【表3】

表3

```

MERGE_SESSION(SID1, SID2, Guarantee_Set);
/* 全部のクライアントがファイルの最新バージョンを持ってい
   ることを、複数クライアントが保証できるようにする。 */
SID3 := Allocate_State();
SID3.GS := Guarantee_Set
SID3.RS := SID1.RS ∪ SID2.RS /*二つの読出し集合の和集合を取る*/
SID3.WS := SID1.WS ∪ SID2.WS;
Return SID3;
END_MERGE_SESSION

```

【0027】表1～3のセッションマネージャの一実施例の概略に示されているように、セッションの基本命令により、クライアントはセッションを開始、終了、格下げ、コピー作成、併合することができる。セッションを「開始」するには、クライアントはセッションマネージャにStart_Sessionを呼び出す。セッションマネージャは、それに応答して、「セッションID」("SID")により固有に識別される「セッション状態」を、割り当てる。セッション状態は、"Read_Set"、"Write_Set"、"Guarantee_Set"といった状態変数を含む。

【0028】"Guarantee_Set"は、クライアントからセッションマネージャに渡されるパラメー

タである。従って、このパラメータは、セッションの最中に実施されるべく指定された所定保証事項のリストである。任意のセッションの"Read_Set"は、セッション中に発生する読出しに「関連した」書込み集合である。この書込み集合は、そのサーバで実施される読出し結果によって指定の保証集合が満たされることを保証することがサーバに知られなくてはならない書込み集合となるように定義される。Read_Setは、セッション中に発生したあらゆる読出しの関連書込み集合の和集合である。Read_Setに含まれる書込みは、クライアントの書込みから、または、別のサーバから生じる。"Write_Set"は、クライアントまたは任意セッションのクライアント関係者から発生する。

【0029】セッションを「終わらせる」には、クライアントはセッションマネージャでEnd_Sessionを呼び出して、終了させるセッションを固有に識別するSIDを渡す。セッションを終わらせるとき、セッションマネージャは、終了されたセッションの状態に対して予め割り当てられていたメモリを単に割り当て解除し、SIDの参照を外す。

【0030】図14は、いくつかのコードルーチン190（例えば、Start_Sessionコード192が供給される）とセッション状態210の記憶領域から成るセッションマネージャのブロック図である。図3は、アプリケーション業務24による具体例としてのセッションを図示したものであり、そのコード32が示されている。セッションマネージャ22は、ライブラリとしてリンクされることによるアプリケーション業務24の一部である場合もあるし、アプリケーション業務とは関係無く、ネームサービスにより探し出され、RPSインフラストラクチャによりアクセスされる別個のプロセスの場合もある。

【0031】アプリケーション業務24は、セッション中に実施したい所定保証事項のリストを渡す。次に、セッションマネージャ22は、セッションを識別するためにセッションID（"G1"）をリターンする。"G1"は、GS（すなわち、保証集合）214、RS（すなわち、Read_Set）216、WS（すなわち、Write_Set）218というフィールドを含むセッション状態210に対するポインタ212として実施される。これらのフィールドは、動的にリンクされるリスト、従って、セッションの最中にフィールドを成長させるリストとして実施される。セッションマネージャの記憶領域は、一度に一つ以上の活動中セッション（例えば、セッションG2 220）を含む。

【0032】セッションの一部である、後続のあらゆる読出しおよび書込みは、存在セッションID、G1を含んでいる。読出しおよび書込みコマンドは、各々、セッションマネージャ190にある読出しおよび書込み手順に対する呼び出しとして実施される。読出しの場合、セッションマネージャはセッションID G1を積極的に取り去り、どのサーバが十分に最新であるか確認し、このようなサーバの少なくとも一つに待ち行列を発行し、その結果をアプリケーション業務に戻し、必要に応じてセッション状態のRSを更新する。書込みの場合、セッションマネージャ190はセッションID G1を取り去り、そのサーバが適切であるか確認し、少なくとも一つの適切なサーバにアップデートを発行し、必要に応じてセッション状態のWSを更新する。

【0033】セッションを終了する場合、アプリケーション業務24はEnd_Sessionコールを出す。次いで、セッションマネージャは、セッションG1に使用されていた記憶領域を回復し、当該特定セッションの

セッションIDの参照を外す。

【0034】前述の実施例の場合、サーバの協同について、三つの仮定が立てられている。

(1) 各クライアントが始めた書込みの場合、サーバはその書込みに固有に割り当てられた、そのシステムで唯一の書込みID（WID）をリターンする。

(2) 各読出しの場合、サーバは、待つ行列の結果と、当該「読出し」に「関係する」WIDの集合をリターンする。

(3) クライアントは、サーバの状態（すなわち、サーバに「知られている」全部の書込みの集合）を獲得できる。

【0035】クライアントの読出しに応じてWIDがサーバからセッションマネージャに戻されると、読出しに関連するWIDがRead_Setに添えられる。同様に、クライアントの書込みに応じて返されるWIDは、Write_Setに添えられる。

【0036】各クライアントは、自分自身の一貫性 対 利用可能性のトレードオフを、システム基模のデザイン選択として享受するのではなく、自由に決定できる。クライアントにとっての「優れた」戦略は、このような保証を最少数選択してクライアントの目的に合わせることであるが、これは、そうすることにより、任意のセッションの最中に出される特定の読出し／書込み要求が、いずれかの適切なサーバによって満足される機会が最大限になるからである。

【0037】一旦、セッションが始まると（そして、最初に保証の部分集合が選択されると）、クライアントはセッションマネージャに対してDowngrade_Guaranteeを呼び出すことによって保証を格下げすることもある。いずれかと与えられた時間内に実施される全ての保証はセッション開始からの実施であるので、保証の格下げは「統合された」セッションという概念を失なわない。

【0038】ユーザならびにアプリケーション業務は、データベースを更新し、その後すぐに、まだ更新されていない別のサーバで読んだために、その更新が消えてしまったように見えることを確認するだけのためにデータベースから読出しを行なうと混乱をきたす危険があるので、「自分の書込み読出し（RYW）」保証が望ましい。この保証は、あるセッション内で行われたいずれか書込みの効果を、そのセッション内の読出しで見られることを保証するものである。具体的にいうと、次の通りである。

【0039】RYW-保証： セッション中に、読出しRが書込みWの次にある場合、Rは時間tにサーバSにて実施され、その後、WはDB（S，t）に収められる。

【0040】アプリケーション業務は、同データ項目の書込みに続く読出しが直前に書き込まれた情報をリター

ンするということを保証されない。これは、読出しはセッション外で実施された（おそらく他のクライアントによる）別の書込みを見る場合もあるからである。

【0041】図4と5は、RYW保証の重要性と効果を説明するものである。図4は、RYWを実施しないで、望ましくない結果が生じる筋書を示す。

【0042】ステップ41： クライアント14a（"C1"）は、\$100から\$150への新残高を反映すべく自分の新預金口座をサーバ12a（S1）にて更新する。

ステップ42： S1は、その変更をサーバ12c（Sn）に「遅延」伝播する。

ステップ43と44： 続いて、C1はサーバ12b（S2）にて自分の口座残高を「読出し」、「古い」預金残高\$100を受け取る。

ステップ45： それからしばらくして、S1はS2へ更新を遅延伝播する。

【0043】図5は、図4と同じ筋書であるが、RYW保証が選択および実施されている点異なる。図4では、クライアントC1が次のプログラムを実行すると仮定する。

```
G=Start Session (RYW);
Write (G, X, 150);
Value=Read (G, X);
End Session (G);
```

【0044】ステップ51： C1はセッションGを「開始」し、新残高\$150をS1に書き込む。

ステップ52： S1はその書込みに対してWIDを割り当てて、C1にリターンする。

ステップ53： C1は、新WIDをそのWrite_Set 30

ステップ54： S1は、現在のデータならびに元の更新要求に関わるWIDを、Snに遅延更新する。

ステップ55： C1は、S1またはSnから、セッションGで読み出さなくてはならないことを「発見」する。

ステップ56： C1はセッションGにてSnから読出しを行う。

ステップ57： S2は新残高\$150をリターンする。

ステップ58： S1は後でS2を更新する。

【0045】ステップ55の「発見」は、C1のセッションマネージャによって実施されるプロセスであって、実施保証に対するクライアントの要求を満足するデータを有する適当なサーバ集合を見つけることである。大まかにいうと、これは、クライアントの現要求に応じて、任意のクライアントが「到達可能な」全部のサーバを捜し出し、そのようなサーバ全部に対して、それらの書込み情報（すなわち、サーバの書込み集合またはDB（S））の照会を行い、サーバの書込み集合とクライア 50

ントのWrite_Setおよび/またはRead_Setを比較して、これらサーバのWrite_SetがクライアントのWrite_Setおよび/またはRead_Setを含んでいるか検査することによって、達成される。いずれのサーバのWrite_Setもこの検査を満足しない場合には、データは「保証」されない。それにも関わらず、クライアントはセッションを終了し、保証に関わり無く、とにかくデータを検索回収することを選択できる。

10 【0046】自分の書込み読出し保証の実施は、二つの基本ステップを必要とする。サーバにより書込みが受け付けられた場合には必ず、セッションの書込み集合にその割り当てWIDが加えられる。サーバSの時間tの各読出しの前に、セッションマネージャは、その書込み集合がDB（S，t）の部分集合であることを確認しなくてはならない。この確認は、クライアントの書込み集合をサーバに渡すことによってサーバで行うことも、サーバのWIDリストを検索回収することによってクライアントで行うことも出来る。セッションマネージャは、チェックが成功したものを発見するまで、利用可能サーバを試し続けることが出来る。適切なサーバが見つからない場合、保証が提供されないことをクライアントに対して報告する。

【0047】2.2.2 「単調読出し」保証（MR）
単調読出し保証では、ユーザは、時間をかけてどんどん最新化するデータベースを観察することができる。単調読出し保証は、セッション内の以前の読出しで書込み効果が目撃された全部の書込みが入っているデータベースコピーに対してのみ、読出し操作が行われることを保証するものである。

【0048】二つの付加定義（「完全」と「関連書込み」）が必要である。直感的にいうと、書込み集合がデータベースの書込みの「十分」を含んでいれば、書込み集合は読出しの結果を「完全」に判断するので、書込み集合に対する読出し実行の結果は、全データベースに対する読出し実行と同じものである。具体的にいうと、WSがDB（S，t）の部分集合である場合、あるいは、部分集合であるだけでも、書込み集合WSは、読出しRとDB（S，t）について完全であり、ならびに、WS 40を含み且つDB（S，t）の部分集合でもあるいずれか集合WS2について、WS2に適用される結果Rは、DB（S，t）に適用される結果Rと同じである。

【0049】「関連書込み（S，t，R）」とは、読出しRとDB（S，t）にとって完全な最小書込み集合をリターンする機能のことである。DB（S，t）は、いずれか読出しについて完全であるから、完全集合は確かに存在する。最小完全集合が固有のものでない場合、結合は、自由裁量であるが決定論的に破壊される。これらの定義から、単調読出し保証は次のように正確に定義で 50

【0050】MR-保証：セッションでR2の前に読出しR1が発生し、R1は時間t1にサーバS1にアクセスし、R2は時間t2にサーバS2にアクセスするならば、関連書込み(S1, t1, R1)は、DB(S2, t2)のサブ集合である。

【0051】図6と7は、一例としてのクライアントの要求でアプリケーションに応じて、MR保証を図で示したものである。図6は、MR保証がない場合の結果を示す。

ステップ61：C1はサーバS1にて、\$100から \$150への新残高を反映するように、自分の預金口座を更新する。

ステップ62：S1は、その変更をサーバSnに「遅延」伝播する。

ステップ63とステップ64：次いでC2は、SnにてC1の口座残高の「読出し」を行って、「新しい」口座残高\$150を受け取る。

ステップ65とステップ66：再びC2は、C1の口座残高を、今度はS2から読出し、「古い」残高\$100を受け取る。

ステップ67：それからしばらくして、S1はS2へ更新を遅延伝播する。

【0052】このように、C2は、後続読出しが少なくとも先程の読出しと同じくらいに最新ものであることを保証することなく、先程読出したデータを読み出すことを許可される。

【0053】図7は、図6と同じ筋書であるが、MR保証が実施されている点が異なる。図7では、クライアントC2が次のプログラムを実行すると仮定する。

G=Start Session (MR) ;

Write (G, X) ;

Value=Read (G, X) ;

End Session (G) ;

【0054】ステップ71と72：C1はセッションGを「開始」し、\$150という新残高をS1に書き込む。S1はその書込みに対する固有のWIDをリターンする。

ステップ73：S1は、WIDと一緒に、更新をSnに遅延伝播する。

ステップ74と75：C2はSnから残高を読み出す。Snは要求に応じて、その結果に関連するWIDと一緒に、残高をリターンする。

ステップ76：C2は、そのセッション状態で格納されているそのReadSetに、WIDを収める。

ステップ77：その後、C2は残高の別のコピーを欲し、S1とSnだけが「十分に」最新なものであることを「発見」する。

ステップ78と79：C2は残高について、S1の読出し要求を行う。S1は、WIDと一緒に残高をリターンする。

ステップ80：S1は後で、WIDと一緒にS2を更新する。

【0055】ステップ77で、S1とSnが「十分に」最新な唯一のサーバであるというC2の発見は、RYW保証について前述されたのと同様な発見プロセスを伴うものである。唯一異なるのは、MR保証は、クライアントの(Write_Setではなく)Read_Setに対してサーバのWIDを比較することである。

【0056】読出し後書込み保証は、全部のサーバにて、伝統的な読出し/書込み依存が、書込み順に保存されることを保証するものである。従って、データベースのあらゆるコピーにおいて、セッション中になされた書込みは、そのセッション中の以前の読出しで効果が目撃された書込みの後に順序付けられる。

【0057】WFR-保証：セッション内で読出しR1が書込みW2に先んじ、R1が時間t1にサーバS1で実施されるならば、いずれかサーバS2について、W2がDB(S2)にある場合、関連書込み(S1, t1, R1)のいずれかW1もDB(S2)と書込み順序(W1, W2)にある。

【0058】前述の二つの保証と違って、この保証は、セッション外のクライアントに影響を及ぼす。以前に目撃されている書込みの後にクライアントの実施する書込みが発生することをセッションが観察するだけでなく、他のすべてのクライアントも、セッション保証を要求しているか否かに関わり無く、同一な書込み順序付けを見る。

【0059】WFR保証は、定義されている通り、書込み操作に関する二つの制約に関連がある。「書込み順序」に関する制約は、結局、全部のデータベースの複製が反映する全体的な順序で、書込みが以前の関連書込みに続くことを保証する。「伝播」に関する制約は、全部のサーバ(従って、全部のクライアントも)は、関わりのある以前の書込み全部を目撃した後に、ある書込みを見るだけであることを保証する。アプリケーション業務の中には、下記の読出し後書込み緩和変更がサポートされるように、これらの制約の一方のみを要求するものもある。

【0060】WFRO-保証(「順序」制約)：セッション内で読出しR1が書込みW2に先んじ、R1が時間t1にサーバS1で実施されるならば、関連書込み(S1, t1, R1)のいずれかW1について、書込み順序(W1, W2)である。

WFRO-保証(「伝播」制約)：セッション内で読出しR1が書込みW2に先んじ、R1が時間t1にサーバS1で実施されるならば、いずれかのサーバS2もついで、W2がDB(S2)にあれば、関連書込み(S1, t1, R1)のいずれかW1もDB(S2)にある。

50 【0061】図8と9は、一例としてのクライアントの

要求に関連したWFR保証を図で示すものである。図8は、WFR保証が実施されない場合の結果を示す。

ステップ81: C1はサーバS1にて、\$100から\$150への新残高を反映するように、自分の預金口座を更新する。

ステップ82: S1は、その変更をサーバSnに「遅延」伝播する。

ステップ83とステップ84: 次いでC2は、SnにてC1の\$150となる口座残高の「読出し」を行う。

ステップ85: C2は新残高\$200をS2に書き込む。 10

ステップ86: S2は\$200という値をSnに伝播する。しかしながら、Snは、残高は\$150のまま、C1からの更新をC2からの更新の後ろに順序付ける。

ステップ87: S1は\$150という値をS2に伝播する。このとき、S2は残高について、C1からの\$150とC2からの\$200という二種類の更新を受け取っている。S2は、値は\$150のまま、C1からの更新を、C2からの更新の後に順序付ける。

ステップ88: S2は、\$200という値をS1に伝播する。今度は、S1は残高について、C1からの\$150とC2からの\$200という二種類の更新を受け取っている。S1は、所定のアルゴリズムを利用して、値は\$150のまま、C1からの更新を、C2からの更新の後に順序付けるが、である。 20

【0062】ステップ87の後、S2の残高の現バージョンは、実際には正しい現残高に先行するバージョンである。

【0063】図9は、図8と同じ筋書であるが、WFR保証が実施されている点が異なる。図9では、クライアントC2が次のプログラムを実行すると仮定する。 30

```
G=Start_Session(WFR);
Value=Read(G,X);
Write(G,X,Value+50);
End_Session(G);
```

【0064】ステップ91と92: C1はセッションGを「開始」し、\$150という新残高をS1に書き込む。S1はその書き込みに対する固有のWIDをリターンする。

ステップ93: S1は、WIDと一緒に、更新をSnに遅延伝播する。 40

ステップ94と95: C2はSnから残高を読み出す。Snは要求に応じて、その結果に関連するWIDと一緒に、残高をリターンする。

ステップ96: C2は、そのセッション状態にて格納されているそのRead_SetにWIDを収める。

ステップ97: その後、C2は残高を更新することを欲し、S1とSnだけが「十分に」最新なものであることを「発見」する。

ステップ98と99: C2は残高について、Snの書 50

込み要求を行う。Snは固有WIDをリターンする。

ステップ100: Snは後で、WIDと一緒に、S1とSnで行われた書き込みを、その書き込み順序で渡すことによって、S2を更新する。

ステップ101: S1は、WIDと一緒に、その更新されたデータを渡すが、S2は、既にこのWIDを見たことと知らせるので、この更新を再処理することはない。

ステップ102: その後、Snはその更新データと固有WIDを利用して、S1を更新する。

【0065】自分の書き込み読出し保証ならびに単調読出し保証と違って、読出し後書き込み保証を実施するには、サーバの挙動に二つの追加制約を課す必要がある。

【0066】制約1: サーバが時間tにクライアントから新しい書き込みW2を受け付けるとき、既にDB(S,t)に入っているいずれかW1について、書き込み順序(W1,W2)が真であることを保証する。

制約2: W2が、時間tにサーバS1からサーバS2へ伝播されるならば、いずれか先に順序付けられたDB(S1,t)内のW1もS2に伝播される。

【0067】これら二つの制約は、いずれかDB(S,t)内の書き込みに対してではなく、セッションの読出し集合または書き込み集合のいずれかが書き込みW1に対して守られなくてはならない。

【0068】単調書き込み保証は、セッション内の或る書き込みは、必ず以前の書き込みの後に続くということを意味するものである。具体的に述べると次の通りである。

【0069】MW-保証: セッション内で書き込みW1が書き込みW2に先んじるならば、いずれかサーバS2について、W2がDB(S2)内にあれば、S1もDB(S2)と書き込み順序(W1,W2)にある。

【0070】この保証は、セッションのユーザならびにセッション外のユーザの両方に関連のある保証を提供する。書き込み後読出し保証でそうであったように、アプリケーション業務が書き込み順序と書き込み伝播を別々に制御できるよう、変更態様を定義することも出来る。

【0071】図10と11は、各々、MW保証が実施されていないクライアントトランザクションと実施されているクライアントトランザクションを示すものである。図10では、次の通りである。

ステップ111: C1はS1にて、\$100から\$150への残高を反映するように、自分の預金口座を更新する。

ステップ112: C1はS2にて、残高\$200を反映するように、同じ預金口座を更新する。

ステップ113: S2は\$200という更新値をSnに「遅延」伝播する。

ステップ114と115: S1は、\$150という値をSnに伝播する。従って、Snは残高について、S1からの\$150とW2からの\$200という二種類の更新を持っている。Snは順序付けアルゴリズムを適用

して、値は\$150のまま、S1からの更新をS2からの更新の後に順序付ける。従って、S1は\$150という値をS2に伝播し、結果として、S2の最終残高は\$150である。

ステップ116: S2は、\$200という値をS1に伝播する。このとき、S1は残高について、C1からの\$150とS2からの\$200という二種類の更新を受け取っている。S1は、値は\$150のまま、C1からの更新を、S2からの更新の後に順序付ける。

【0072】ステップ114と115で、システムは、古いデータ値が伝播されサーバのデータベースに組込まれる、すなわち、新しい方の更新を上書きすることを「許容した」ため、データベースに非一貫性が入り込む。

【0073】図11は、図10と同じ筋書であるが、MR保証が実施されている点異なる。図11では、クライアントC2が次のプログラムを実行すると仮定する。

G=Start_Session (MR);

Write (G, X, 150);

Write (G, X, 200);

End_Session (G);

【0074】ステップ121、122、123: C1は\$150という新残高をS1に書き込む。S1はその書き込みに対する固有のWIDをリターンする。C1は、そのWrite_SetにWIDを収める。

ステップ124: C1は、再び残高を更新しようとし、他のサーバが更新されていないので、S1に書き込まなくてはならない、ということを発見する。

ステップ125、126、127: C2は新残高\$200を書き込む。S1は、その書き込みに対する固有WIDをリターンする。C1はセッションのそのWrite_SetにWIDを収める。

ステップ128: S1は、二種類の更新を、それらのWIDと一緒に、Snに伝播する。

ステップ129: S1は、二種類の更新を、それらのWIDと一緒に、S2に伝播する。

【0075】単調書き込み保証は、読出し後書き込み保証のものと同じサーバ挙動に関する制約を要する。これらの制約が適所にあれば、単調読出しの実施は更に二つのステップを含む。サーバSが時間tに書き込みを受け付けるには、サーバのデータベースDB (S, t)は、セッションの書き込み集合を含んでいなくてはならない。また、サーバによって書き込みが受け付けられる場合には、必ず、それに割り当てられたWIDが、その書き込み集合に加えられなくてはならない。

【0076】バージョンベクトルは、四種類の保証の更に効果的な実施を考慮するものである。バージョンベクトルは、〈サーバ、クロック〉という順序の対であり、各サーバに一組が割り当てられる。サーバの識別は、単に、複製データベースの特定コピーに対する固有識別子

である。クロックは、人のサーバの単調に増加する論理クロックの値である。この論理「クロック」に関する唯一の制約は、サーバに受け付けられた各書き込みの間に増加しなくてはならない、ということである。〈サーバ、クロック〉対はWIDとしてうまく役立つ。このセクションでは、かかるWIDは最初に書き込みを受け付けたサーバによって割り当てられる、と仮定する。

【0077】各サーバは、次の不変事項により、それ自身のバージョンベクトルを維持する。サーバのバージョンベクトルが〈S, c〉であれば、サーバは、サーバSのクロックの論理時間c以前にサーバSによってWIDが割り当てられた全部の書き込みを受け取っている。この不変事項を維持するために、サーバは遅延伝播のあいだ、割当てWID順に書き込みを転送しなくてはならない。サーバのバージョンベクトルは遅延伝播プロセスの一部として更新されるので、バージョンベクトルはそのデータベース内の書き込み集合を正確に指定する。

【0078】次のように、WID集合がバージョンベクトルと交換される、幾分かもっと実質的な保証を実施することが可能である。

【0079】1) WID集合のコンパクトに表すバージョンベクトルVを求めるには、 $V[S] = (W$ でサーバSによって割り当てられた最新WIDの時間(Sから書き込みがない場合は0))、と置く。

2) 二つのWIDの集合、Ws1とWs2の和集合を表すバージョンベクトルVを求めるには、まず、上記のように、Ws1からV1、Ws2からV2を求める。次に、全部のSについて、 $V[S] = \text{MAX}(V1[S], V2[S])$ と置く。

3) WIDの一方の集合Ws1が、他方のWs2の部分集合か否かを確認するには、まず、上記のように、Ws1からV1、Ws2からV2を求める。次に、V2がV1を優越しているかチェックして確認するが、この場合、優越とは、各対応構成要素において、一方のベクトルが他方のベクトル以上であること、と定義される。

【0080】図15は、十分に「最新な」サーバ選集でのバージョンベクトルの利用を示す。図示の通り、セッションG1 212について維持されている状態は、二つのバージョンベクトルから成っている。一方は、セッションの書き込みを記録するためのもの(WV 22 4')であり、他方は、セッションの書き込みを記録するためのもの(RV 22 2')である(すなわち、セッションの書き込み「関連のある」書き込み)。受け付け可能なサーバを見つけるには、セッションマネージャは、これらのセッションベクトルの一方または両方がサーバのバージョンベクトルによって優越されていることを、チェックしなくてはならない。どちらのセッションベクトルをチェックするかは、セッション内で実施される操作と提供される保証によって異なる。図15の場合、S1 (12a)のバージョンベクトルのみが、セッション

G1のRV. 222'を優越している。他のサーバからS1が「見た」書込みは、少なくとも、セッションG1が見た書込みと同じくらい最新なものである。Sk(12b)のバージョンベクトルは、S1の書込みを、セッションG1と同じくらい多く見てはいないので、優越してはいない。

【0081】サーバは読出しの結果に基づいてバージョンベクトルをリターンして、関連書込みを示す。実際的には、サーバは一般に削除されたデータベースエントリを記憶していない。サーバは、バージョンベクトルに基づいてデータベースのコピーを記憶しているだけである。従って、サーバは、その現バージョンベクトルを関連書込みの総評価として戻せるのである。これは、単調読出し保証や書込み後読出し保証を損なうものではない。受け付け可能なサーバを選択したときにセッションマネージャを過度に保守的にさせるだけである。

【0082】セッションマネージャにおける保証の実施は、図12と13に示されている。いずれか任意のセッションの最中に出される各々の読出しおよび書込み要求は、この手順に基づいてされる。基本的には、この手順は、利用可能な(すなわち、到達可能な)サーバの全体集合で開始して(ステップ150)、どの保証が選択されたかにより、セッションRead_Set(またはWrite_Set)が、任意のサーバSで発生した書込み(すなわち、DB(S))と比較される(ステップ160と166)。セッションRead_Set(またはWrite_Set)がDB(S)に無いならば、Sは、クライアントの要求を処理するのに適したサーバではない(すなわち、その保証はそのサーバに合わない)(ステップ162)。適切なサーバがあったなら(ステップ168)、そのサーバで読出し(または書込み)要求が処理される(各々、ステップ170以降または176以降)。

【0083】バージョンベクトル実施を達成するには、この実施手順に対して、僅かながら変更が必要である。ステップ160: サーバのバージョンベクトルがセッションの読出しベクトルを「優越している」か、チェックする。

ステップ166: サーバのバージョンベクトルがセッションの書込みベクトルを「優越している」か、チェックする。

ステップ172: セッションの読出しベクトル:=Max(読出しベクトル, 関連書込みベクトル)

ステップ178: セッションの書込みベクトル[S]:=WID. クロック

【0084】以前にコンタクトされたサーバは、次の読出しまたは書込み操作を実施するサーバの許容選択選肢である。従って、セッションマネージャが任意のサーバから「離れない」場合には、チェックを飛ばすことが出

来る。セッションマネージャが別のサーバに切り換える場合にのみ、サーバの現バージョンベクトルをセッションのベクトルと比較しなくてはならない。十分に細心なサーバを発見するのを簡単にするために、セッションマネージャは種々サーバのバージョンベクトルを貯蔵できる。サーバのデータベースは、サーバが受け取って組込んだ多数の書込みに基づいて時間をかけて成長できるのみであるので、貯蔵されたバージョンベクトルは、サーバの知識の下限を表すものである。

【図面の簡単な説明】

【図1】 クライアント/サーバ環境の図である。

【図2】 クライアントとサーバとセッションマネージャの構成の一例を示す図である。

【図3】 クライアントの申し込みによって引き起こされた単純なセッションの一例を示す図である。

【図4】 「自分の書込み読出し」保証が不在の場合のクライアントトランザクションの一例を示す図である。

【図5】 「自分の書込み読出し」保証が存在する場合のクライアントトランザクションの一例を示す図である。

【図6】 「単調読出し」保証が課せられない場合のクライアントトランザクションを示す図である。

【図7】 「単調読出し」保証が課せられる場合のクライアントトランザクションを示す図である。

【図8】 「読出し後書込み」保証が課せられない場合のクライアントトランザクションを示す図である。

【図9】 「読出し後書込み」保証が課せられる場合のクライアントトランザクションを示す図である。

【図10】 「単調書込み」保証が課せられない場合のクライアントトランザクションを示す図である。

【図11】 「単調書込み」保証が課せられる場合のクライアントトランザクションを示す図である。

【図12】 セッションマネージャによって実行される発見ならびに実施プロセスのフローチャートである。

【図13】 セッションマネージャによって実行される発見ならびに実施プロセスのフローチャートである。

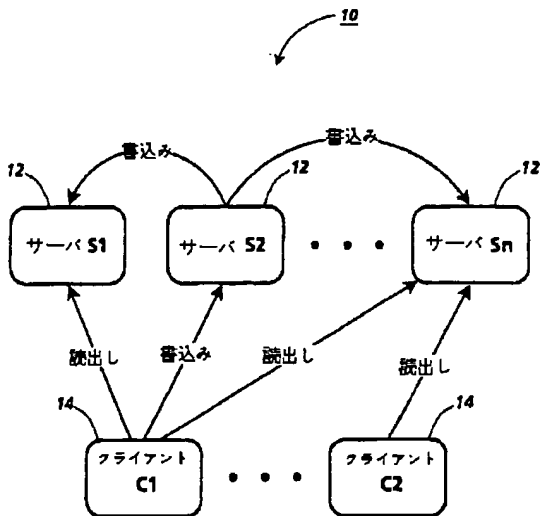
【図14】 セッションマネージャのブロック図である。

【図15】 サーバが十分に最新情報に接しているかを判定するバージョンベクトルの使用法を示す図である。

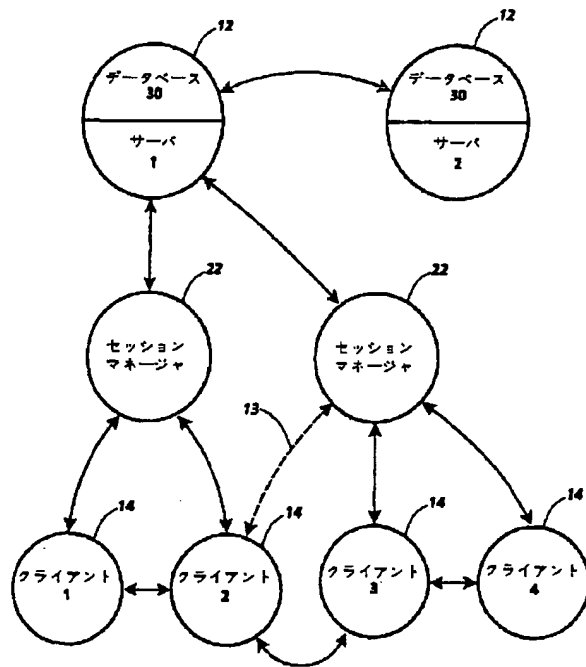
【符号の説明】

10 モデル、12 サーバ、14 クライアント、22 セッションマネージャ、24 アプリケーション業務、30 データベース、32 コード、190コードルーチン、192 Start_Sessionコード、210 セッション状態、212 ポインタ、214 保証集合、216 Read_Set、218 Write_Set

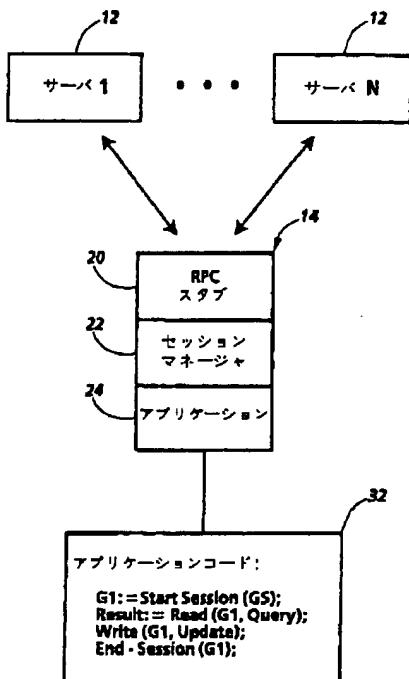
【図 1】



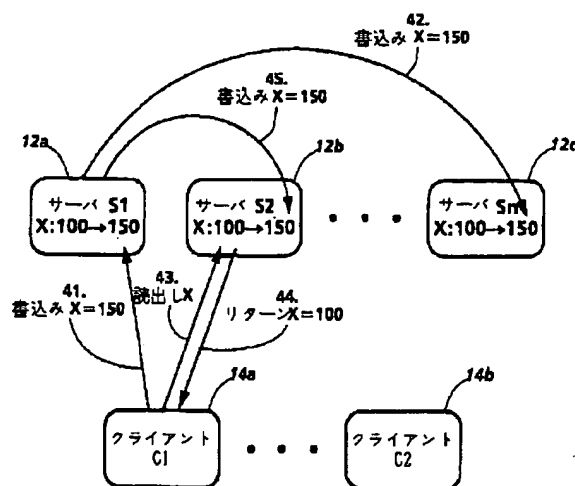
【図 2】



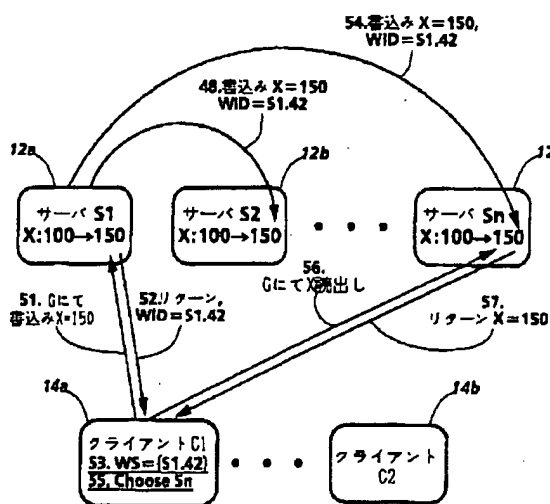
【図 3】



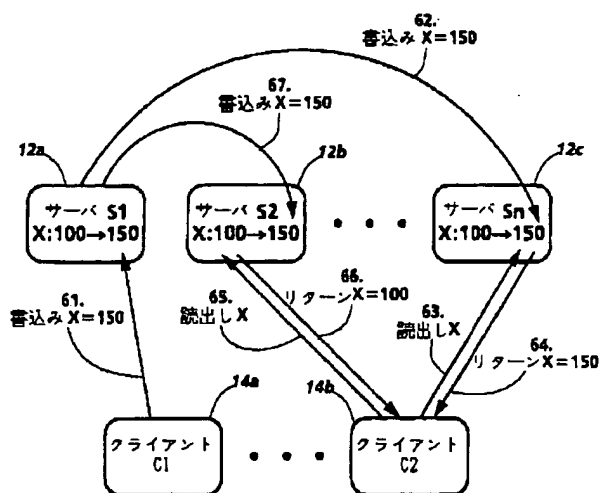
【図 4】



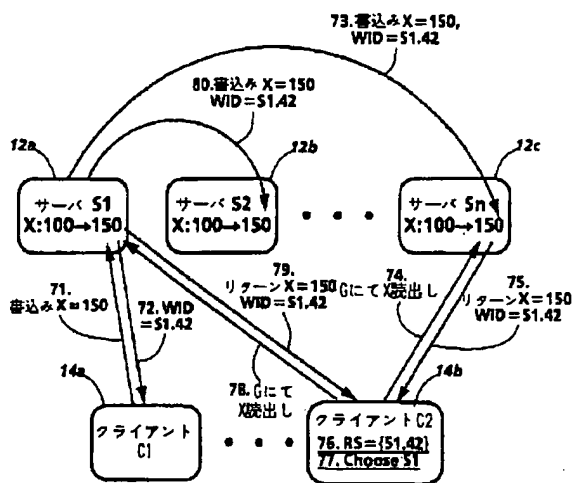
【図5】



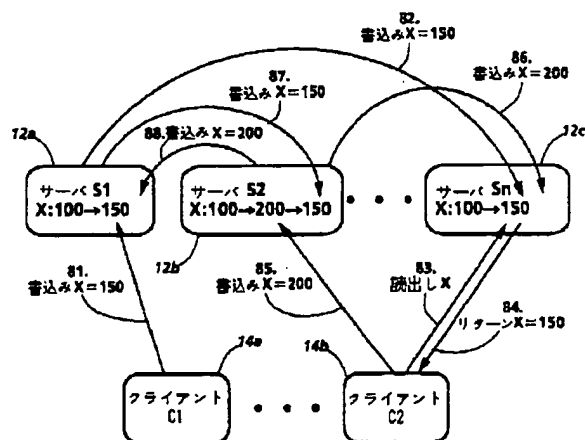
【図6】



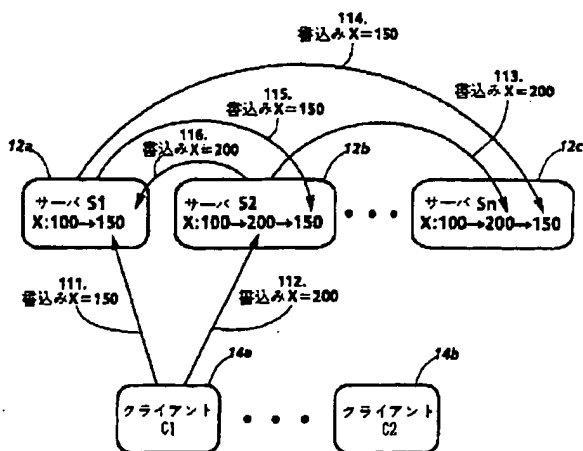
【図7】



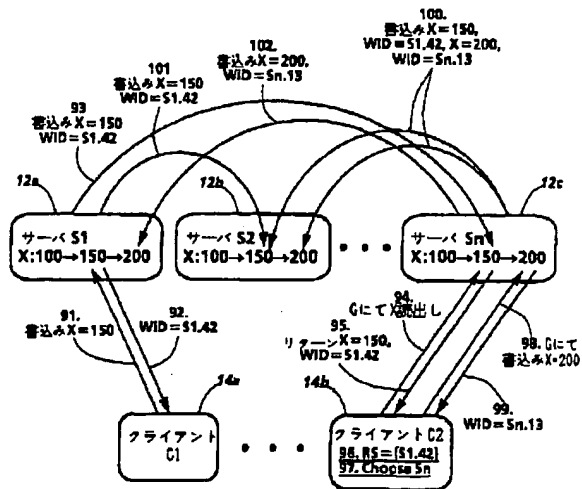
【図8】



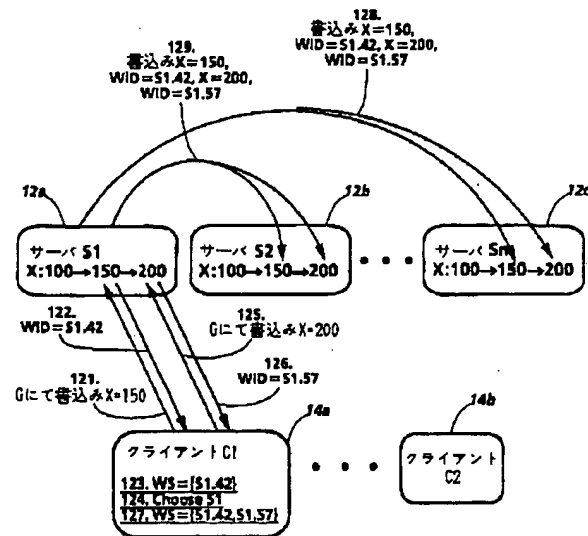
【図10】



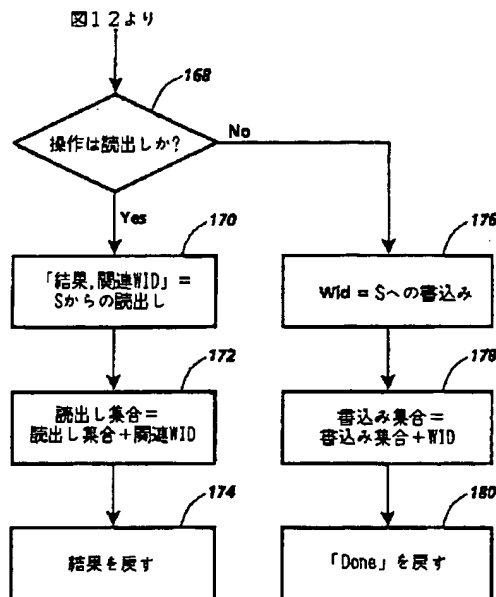
【図 9】



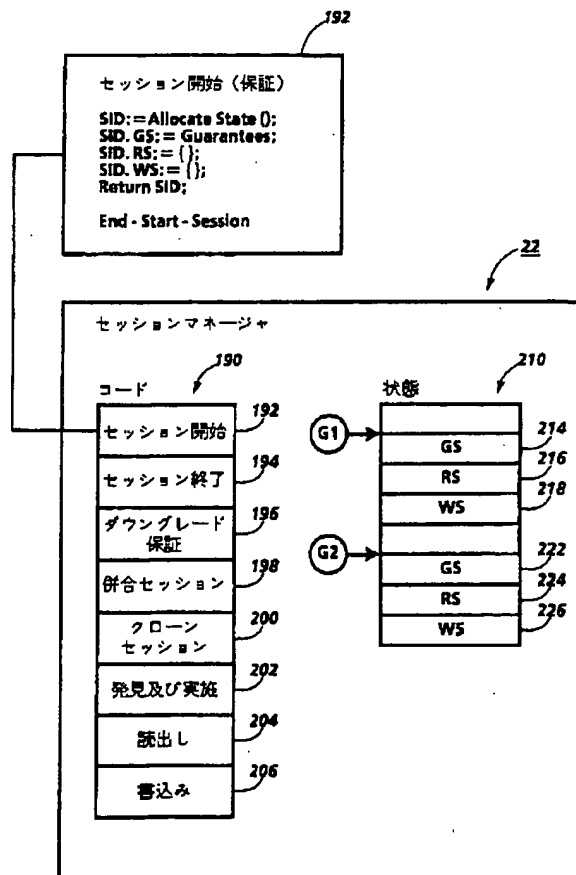
【図 1 1】



【图 13】



【图 14】



【図12】

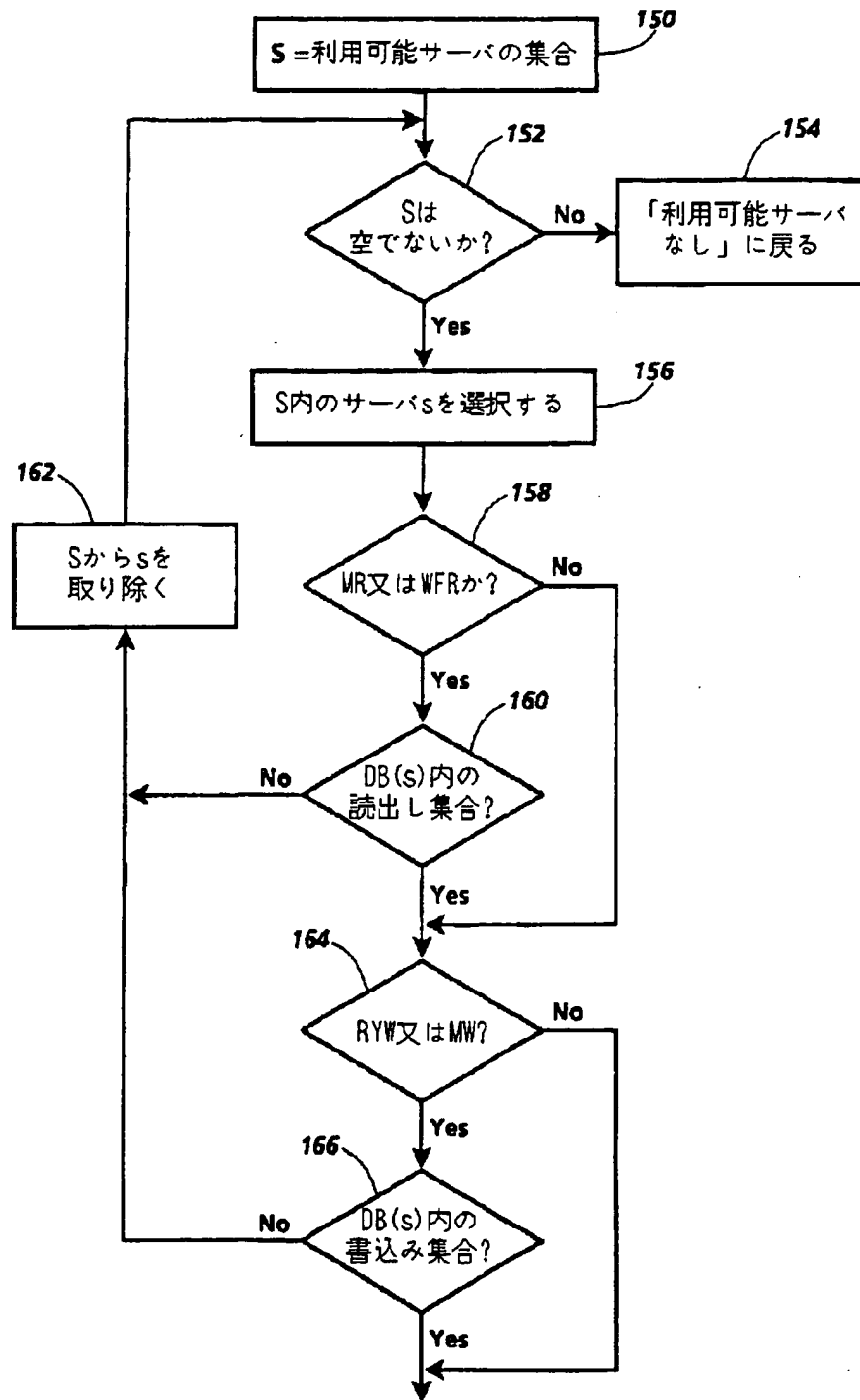
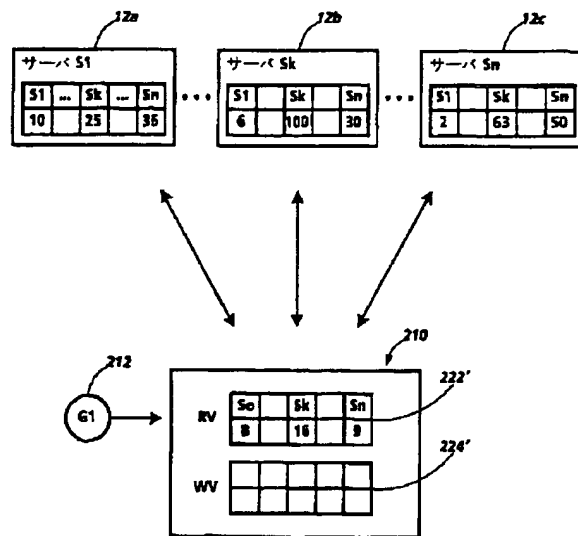


図13へ

【図15】



フロントページの続き

(72)発明者 アラン・ジェイ・ディマース
アメリカ合衆国 カリフォルニア州
95006 プールダークリーク ホプキンス
ガルチ 720

(72)発明者 カリン・ピーターセン
アメリカ合衆国 カリフォルニア州
94025 メンロパーク シャロンパークド
ライブ #エヌ307 350

(72)発明者 マイケル・ジェイ・スプライツァー
アメリカ合衆国 カリフォルニア州
95376 トレイシー プリストルコーンド
ライブ 1941

(72)発明者 マービン・エム・サイマー
アメリカ合衆国 カリフォルニア州
94043 マウンテンビュー アルビンスト
リート 2447

(72)発明者 プレント・ビー・ウェルチ
アメリカ合衆国 カリフォルニア州
94043 マウンテンビュー デルアベニュー
ー 2540